

FIG. 1

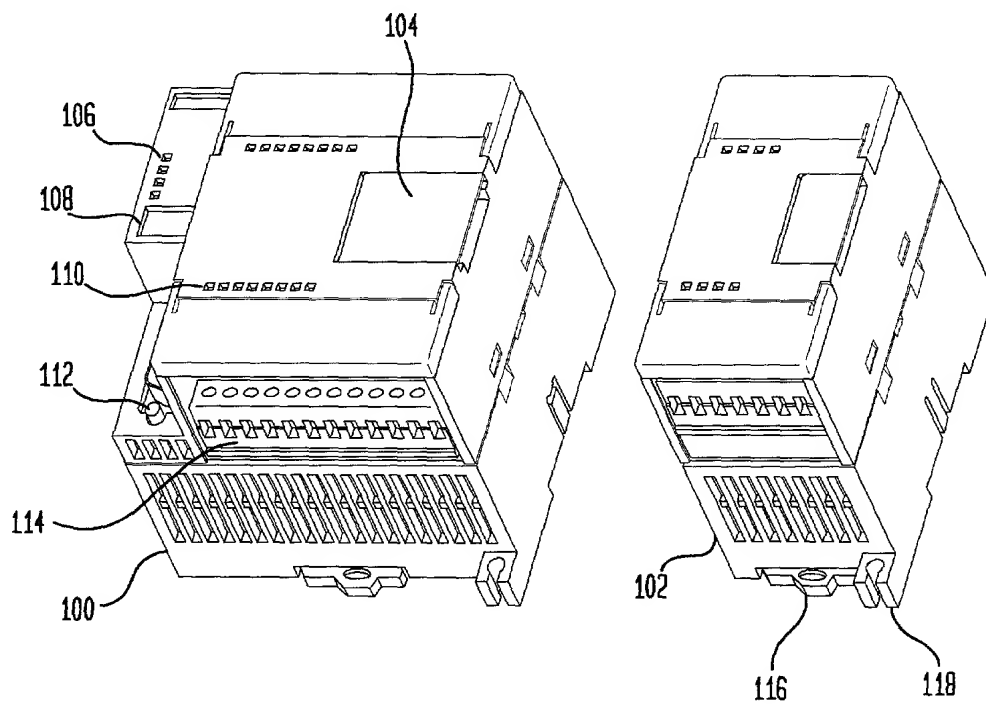


FIG. 2

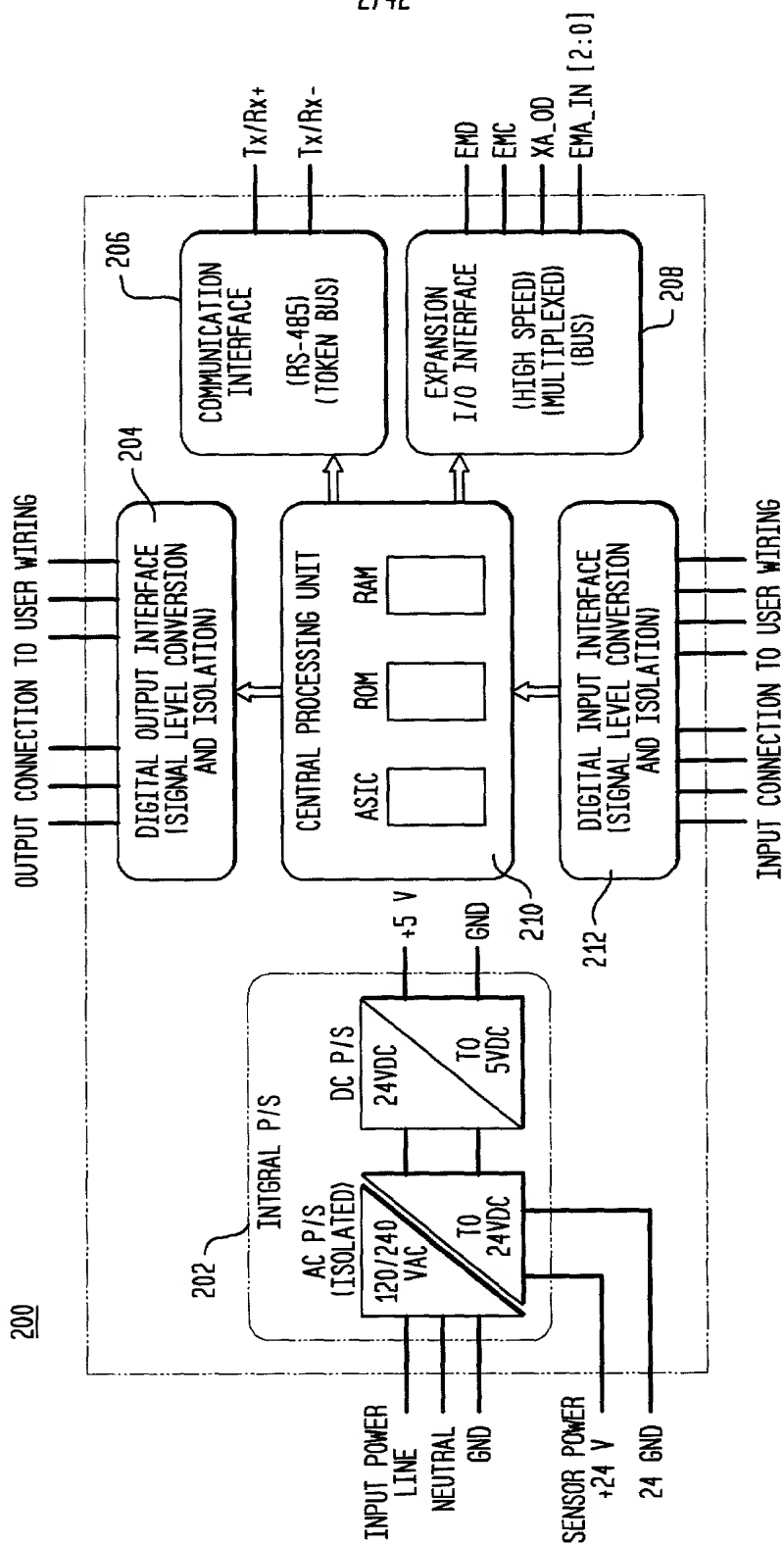


FIG. 3

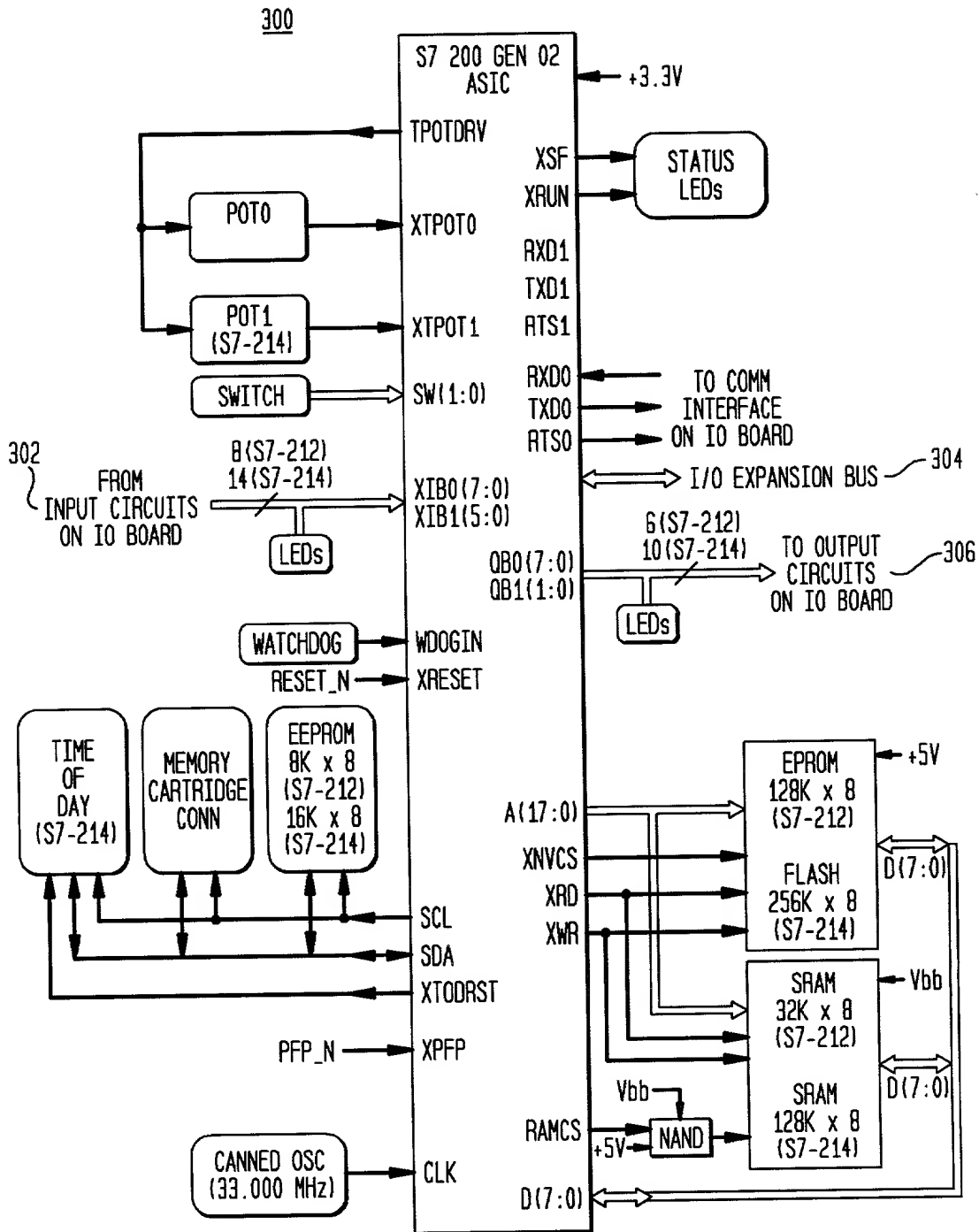


FIG. 4

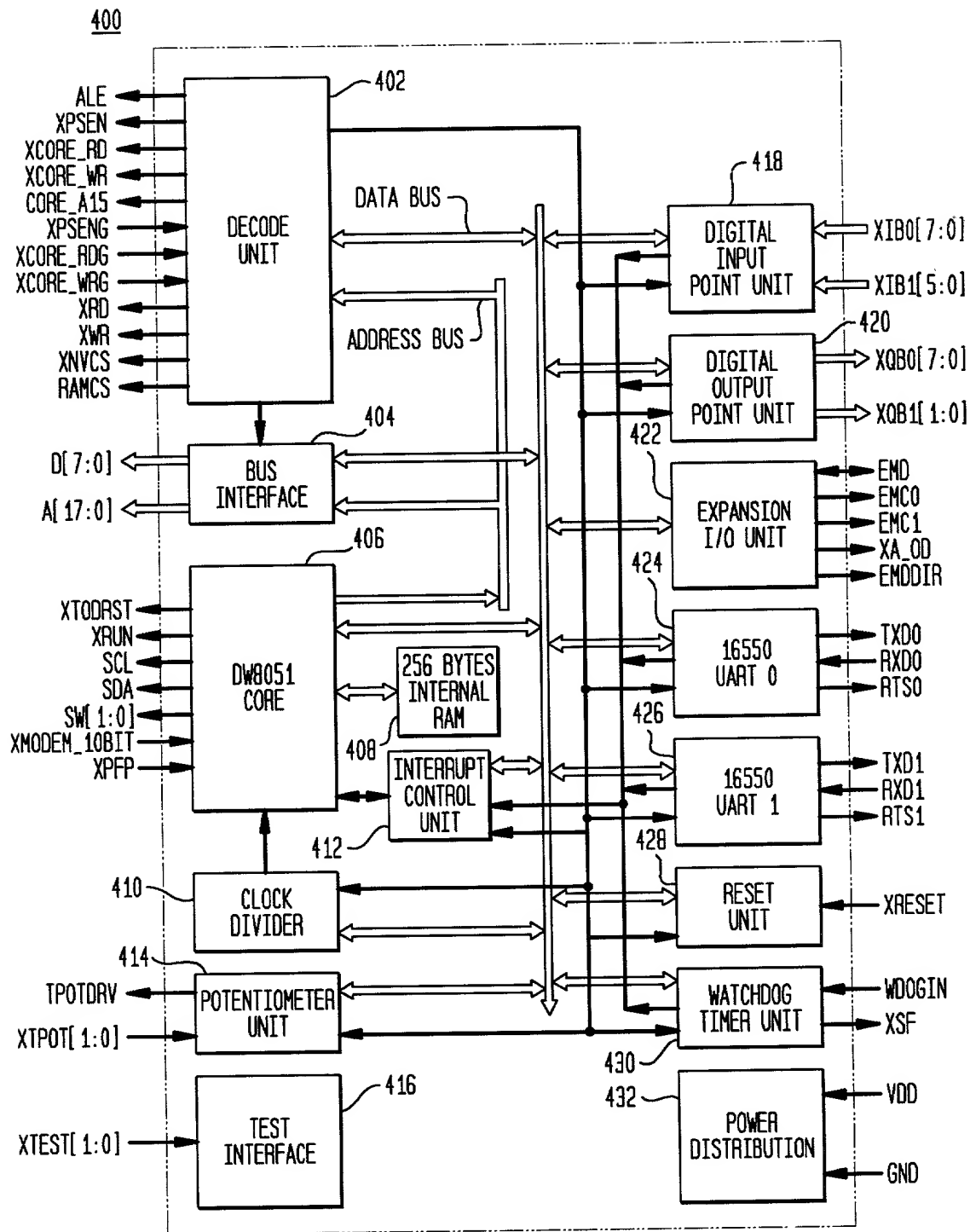


FIG. 5

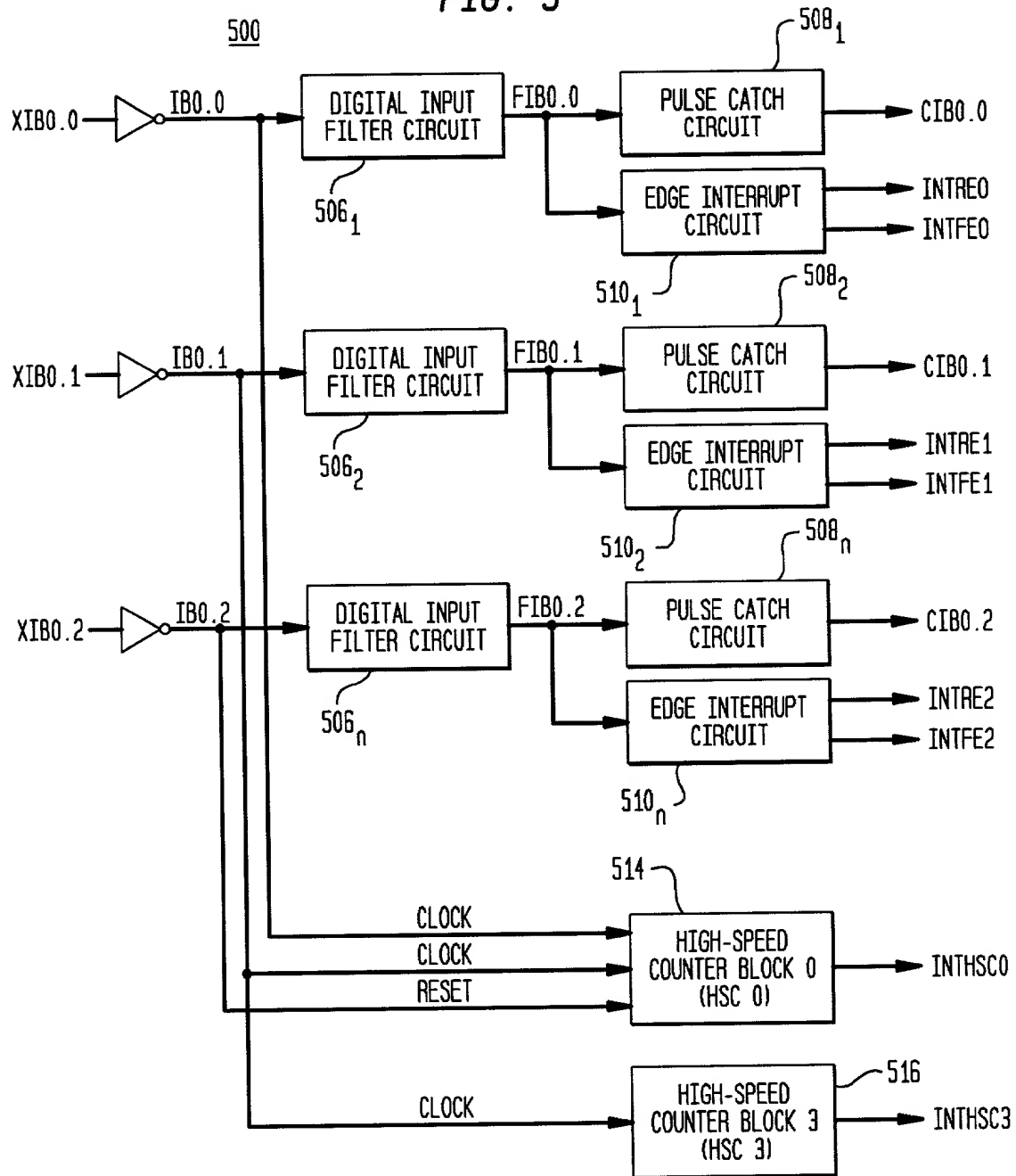


FIG. 6A

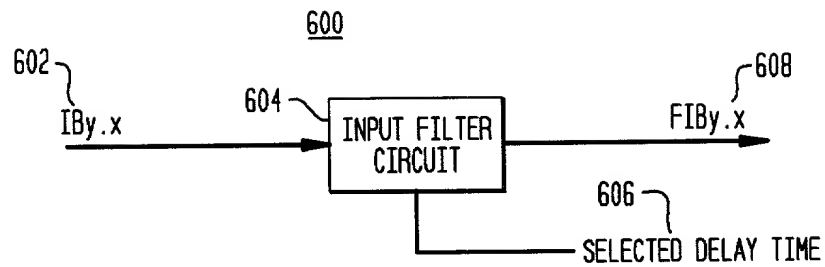


FIG. 6B

DIGITAL FILTER OPERATION DEFINITION

INPUT POINT STATE	CURRENT COUNT	NEXT COUNT	PRESENT OUTPUT VALUE	NEXT OUTPUT VALUE
0 (DECREMENTS COUNTER)	0	0	0	0
	n , where $3 \geq n > 0$	$n - 1$	0	0
	4	3	1	0
	n , where $15 \geq n > 4$	$n - 1$	1	1
1 (INCREMENTS COUNTER)	n , where $11 > n \geq 0$	$n + 1$	0	0
	11	12	0	1
	n , where $14 \geq n > 11$	$n + 1$	1	1
	15	15	1	1

FIG. 6C

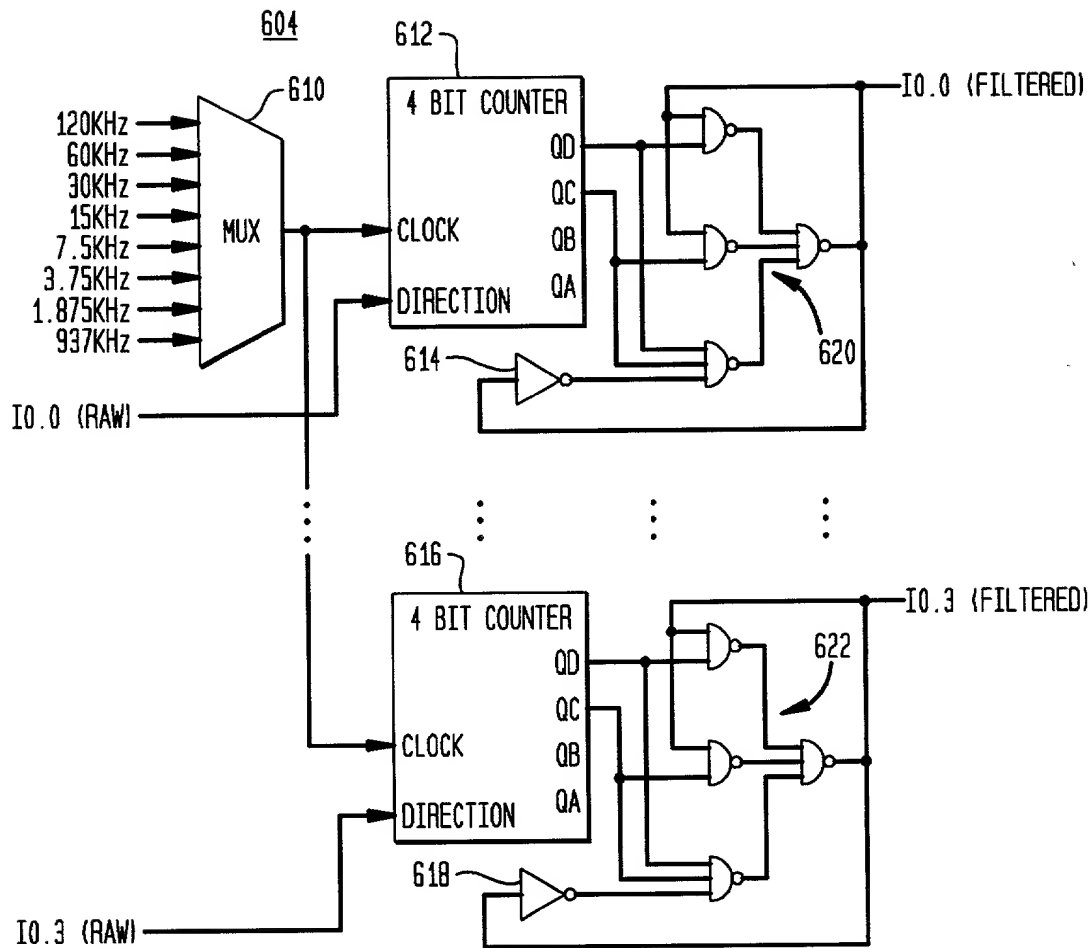


FIG. 6D

FREQUENCY	PERIOD	NUMBER OF COUNTS	DELAY TIME
120 KHz	8.33 μ SEC	12	0.1 ms
60 KHz	16.6 μ SEC	12	0.2 ms
30 KHz	33.3 μ SEC	12	0.4 ms
15 KHz	66.7 μ SEC	12	0.8 ms
7.5 KHz	133 μ SEC	12	1.6 ms
3.75 KHz	267 μ SEC	12	3.2 ms
1.875 KHz	533 μ SEC	12	6.4 ms
937 KHz	1067 μ SEC	12	12.8 ms

FIG. 7

REGISTER VALUE	CORRESPONDING DELAY TIME
00	0.2 ms
01	0.4 ms
02	0.8 ms
03	1.6 ms ¹
04	1.6 ms ¹
05	3.2 ms
06	6.4 ms
07	12.8 ms
08 TO FF	NO DELAY

FIG. 8

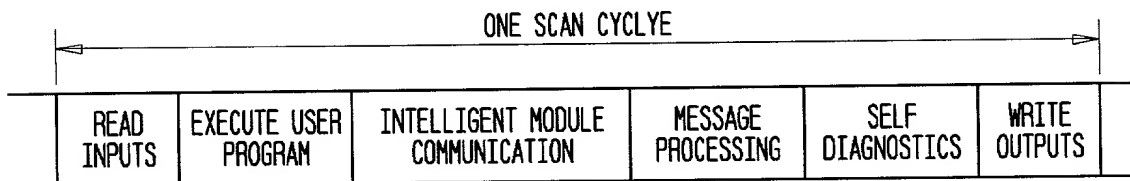


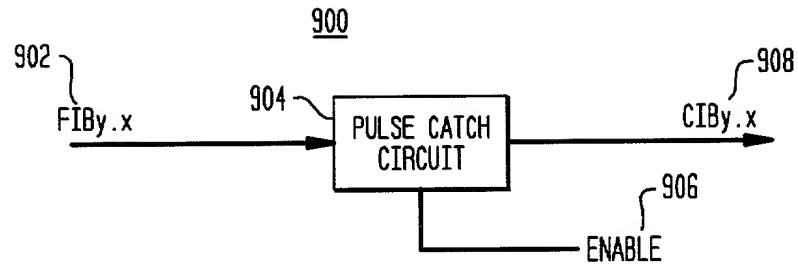
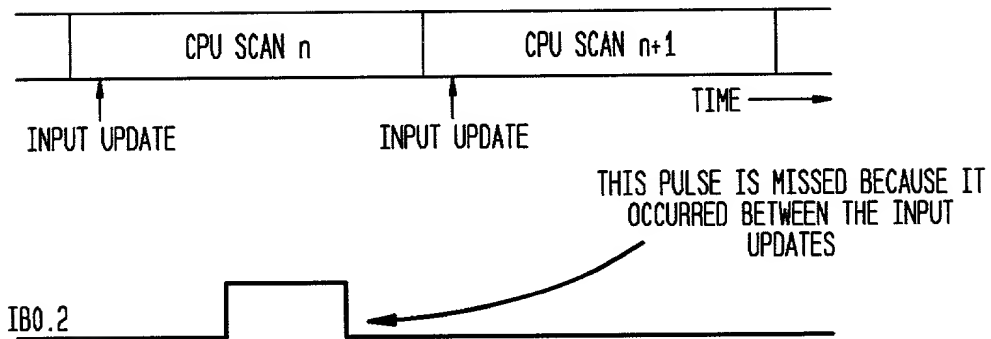
FIG. 9**FIG. 10**

FIG. 11

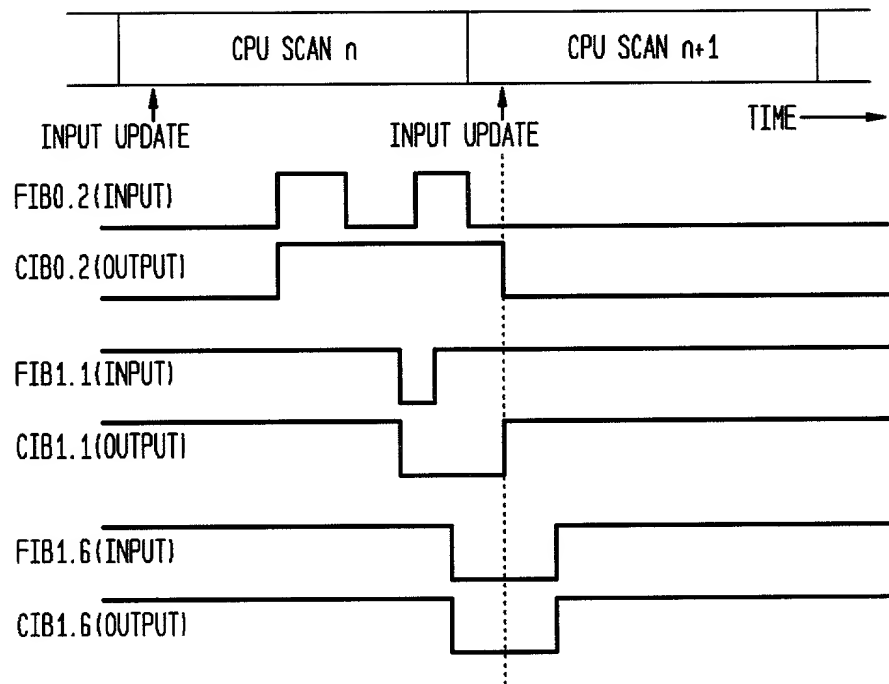


FIG. 12A

PCE	PS CV	PS F	RP	NS CV	NS F	COMMENT
1	I	0	-	CV	0	I = CV AND F IS NOT SET; RP IS A DON'T CARE
1	not I	0	-	I	1	I ≠ CV; CAPTURE NEW VALUE OF I AND SET F = 1
1	-	1	0	CV	1	CV HAS NOT BEEN READ
1	-	1	1	I	0	CV HAS BEEN READ; SET CV = I
0	-	-	-	I	0	PULSE CATCH DISABLED; CV = I

FIG. 12B

PULSE CATCH ENABLE REGISTERS									
ADDRESS	DESCRIPTION								
0002H	REGISTER NAME: IB0_Pulse_Catch_Enable_Register (IB0PCE) SIZE: byte (8-bit) ACCESS: read/write RESET VALUE: 00H 7 0 <table><tr><td>EN7</td><td>EN6</td><td>EN5</td><td>EN4</td><td>EN3</td><td>EN2</td><td>EN1</td><td>EN0</td></tr></table> ENx: 1 = ENABLES PULSE CATCH OPERATION ON INPUT POINT IB0.x 0 = DISABLES PULSE CATCH OPERATION ON INPUT POINT IB0.x	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0		
0003H	REGISTER NAME: IB1_Pulse_Catch_Enable_Register (IB1PCE) SIZE: byte (8-bit) ACCESS: read/write RESET VALUE: 00H 7 0 <table><tr><td>x</td><td>x</td><td>EN5</td><td>EN4</td><td>EN3</td><td>EN2</td><td>EN1</td><td>EN0</td></tr></table> ENx: 1 = ENABLES PULSE CATCH OPERATION ON INPUT POINT IB1.x 0 = DISABLES PULSE CATCH OPERATION ON INPUT POINT IB1.x	x	x	EN5	EN4	EN3	EN2	EN1	EN0
x	x	EN5	EN4	EN3	EN2	EN1	EN0		

FIG. 12C

REGISTER IBOPS:	READ OF THIS REGISTER RETURNS CIB0[7:0] AND RETRIGGERS PULSE CATCH CIRCUITS FOR IB0 INPUT POINTS	USED BY SW FOR INPUT UPDATE
REGISTER IB1PS:	READ OF THIS REGISTER RETURNS CIB1[5:0] AND RETRIGGERS PULSE CATCH CIRCUITS FOR IB1 INPUT POINTS	USED BY SW FOR INPUT UPDATE
REGISTER IBOPSNR:	READ OF THIS REGISTER RETURNS CIB0[7:0] AND LEAVES PULSE CATCH CIRCUITS UNAFFECTED	USED BY SW FOR IMMEDIATE ACCESS
REGISTER IB1PSNR:	READ OF THIS REGISTER RETURNS CIB1[5:0] AND LEAVES PULSE CATCH CIRCUITS UNAFFECTED	USED BY SW FOR IMMEDIATE ACCESS

FIG. 12D

INPUT POINT STATUS REGISTERS															
ADDRESS	DESCRIPTION														
0004H	REGISTER NAME: IB0_Input_Point_Status_Register (IB0PS)														
	SIZE: byte (8-bit)														
	ACCESS: read only														
	RESET VALUE: 00H														
<div><div>7</div><div>0</div><table><tr><td>CI7</td><td>CI6</td><td>CI5</td><td>CI4</td><td>CI3</td><td>CI2</td><td>CI1</td><td>CI0</td></tr></table></div>								CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0
CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0								
CIx: CONTAINS CONDITIONED INPUT POINT STATE CIB0.x															
0005H	REGISTER NAME: IB1_Input_Point_Status_Register (IB1PS)														
	SIZE: byte (8-bit)														
	ACCESS: read only														
	RESET VALUE: 00H														
<div><div>7</div><div>0</div><table><tr><td>x</td><td>x</td><td>CI5</td><td>CI4</td><td>CI3</td><td>CI2</td><td>CI1</td><td>CI0</td></tr></table></div>								x	x	CI5	CI4	CI3	CI2	CI1	CI0
x	x	CI5	CI4	CI3	CI2	CI1	CI0								
CIx: CONTAINS CONDITIONED INPUT POINT STATE CIB1.x.															
0006H	REGISTER NAME: IB0_Input_Point_Status_Register_No_Retrigger (IB0PSNR)														
	SIZE: byte (8-bit)														
	ACCESS: read only														
	RESET VALUE: 00H														
<div><div>7</div><div>0</div><table><tr><td>CI7</td><td>CI6</td><td>CI5</td><td>CI4</td><td>CI3</td><td>CI2</td><td>CI1</td><td>CI0</td></tr></table></div>								CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0
CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0								
CIx: CONTAINS CONDITIONED INPUT POINT STATE CIB1.x															
0007H	REGISTER NAME: IB1_Input_Point_Status_Register_No_Retrigger (IB1PSNR)														
	SIZE: byte (8-bit)														
	ACCESS: read only														
	RESET VALUE: 00H														
<div><div>7</div><div>0</div><table><tr><td>x</td><td>x</td><td>CI5</td><td>CI4</td><td>CI3</td><td>CI2</td><td>CI1</td><td>CI0</td></tr></table></div>								x	x	CI5	CI4	CI3	CI2	CI1	CI0
x	x	CI5	CI4	CI3	CI2	CI1	CI0								
CIx: CONTAINS CONDITIONED INPUT POINT STATE CIB0.x.															

1999P07535US04

FIG. 12E

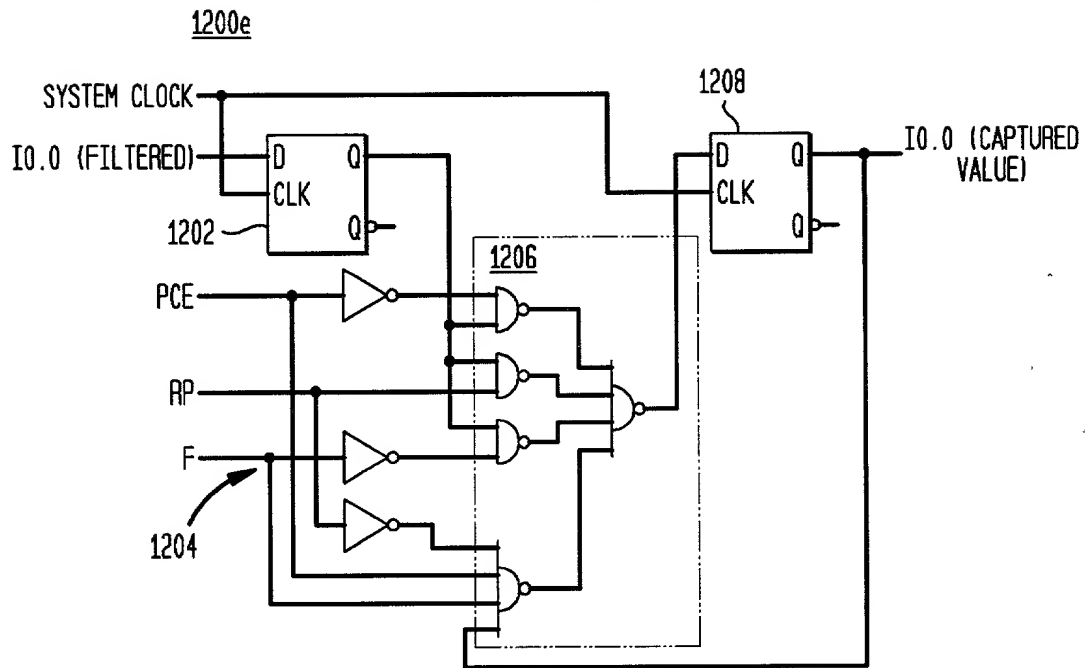


FIG. 12F

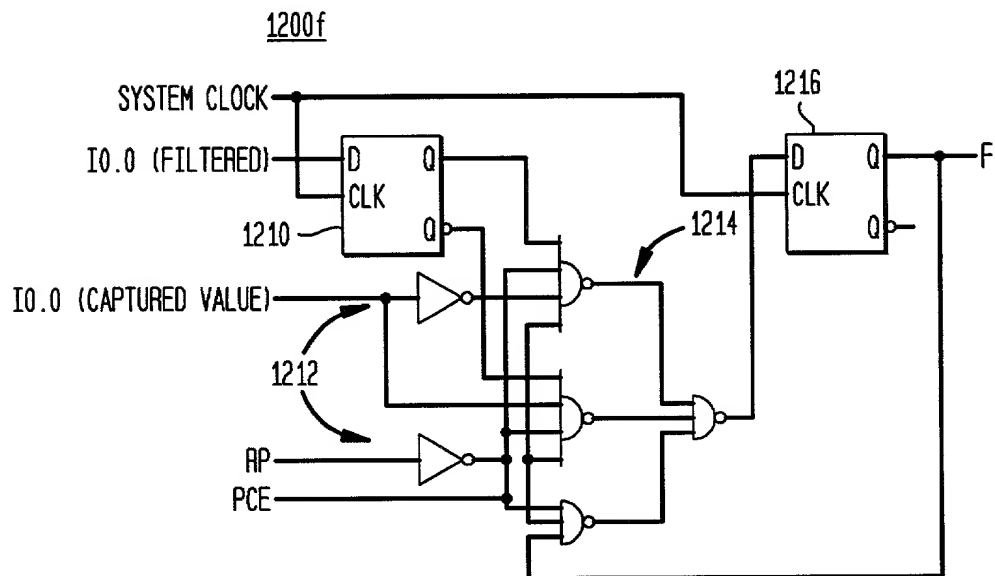


FIG. 13

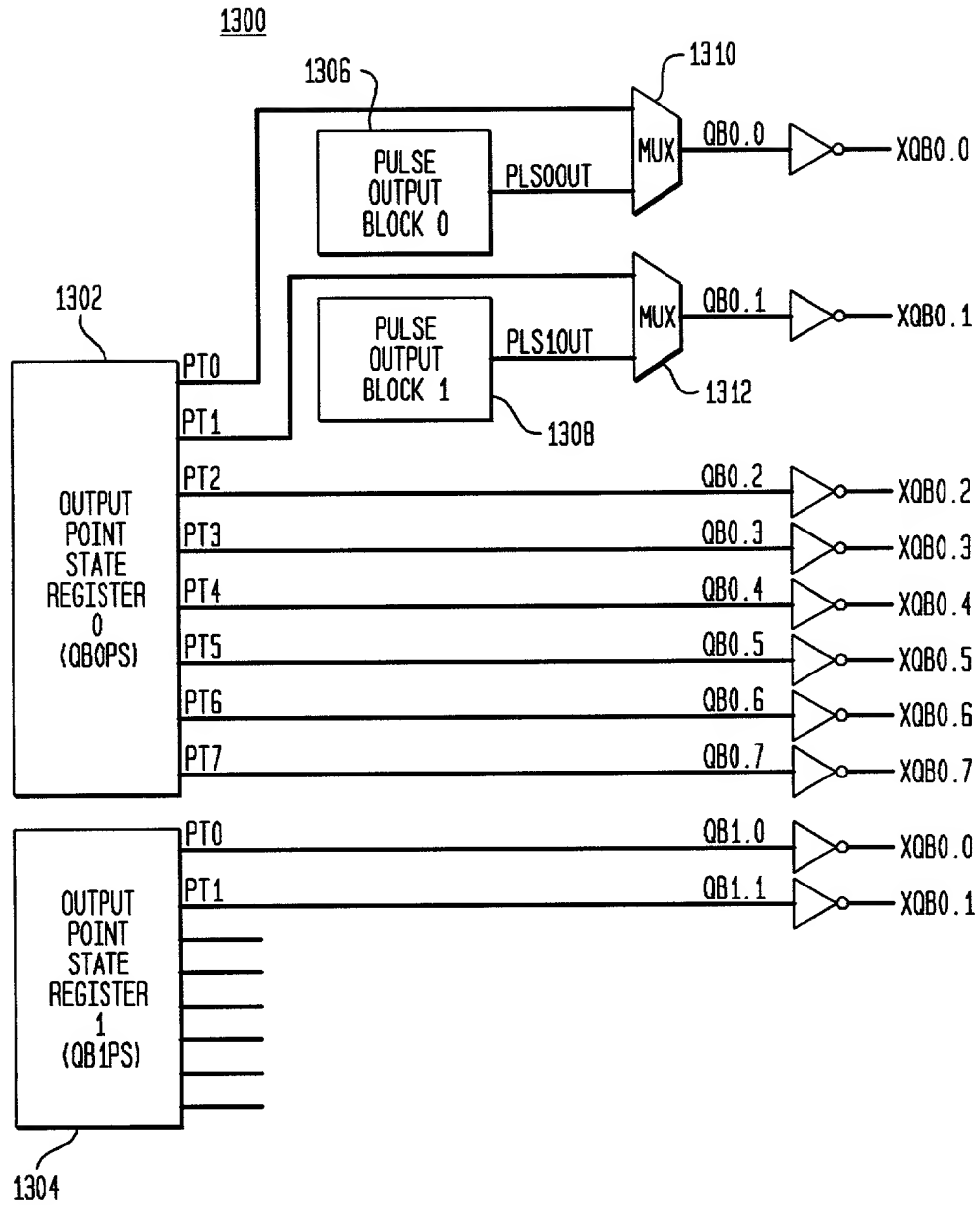


FIG. 14A

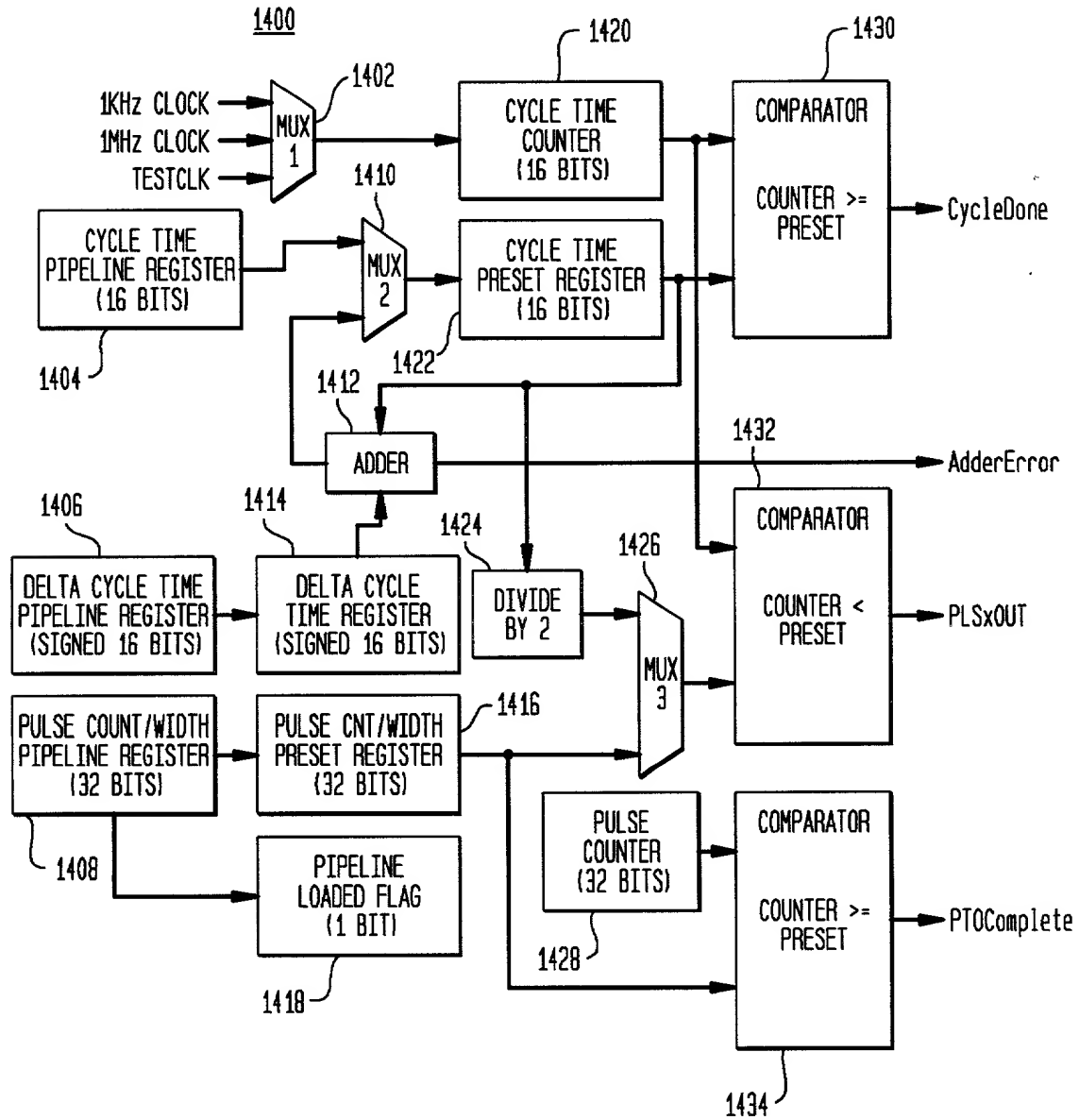


FIG. 14B

REGISTER NAME	VALID VALUE RANGE
CYCLE TIME PRESET REGISTER	2 TO 65535
CYCLE TIME PIPELINE REGISTER	
DELTA CYCLE TIME REGISTER	-32768 TO 32767
DELTA CYCLE PIPELINE REGISTER	
PULSE COUNT/WIDTH PRESET REGISTER	1 TO $(2^{32}-1)$ 0 TO 65535
PULSE COUNT/WIDTH PIPELINE REGISTER	PTO MODE PWM MODE

FIG. 15

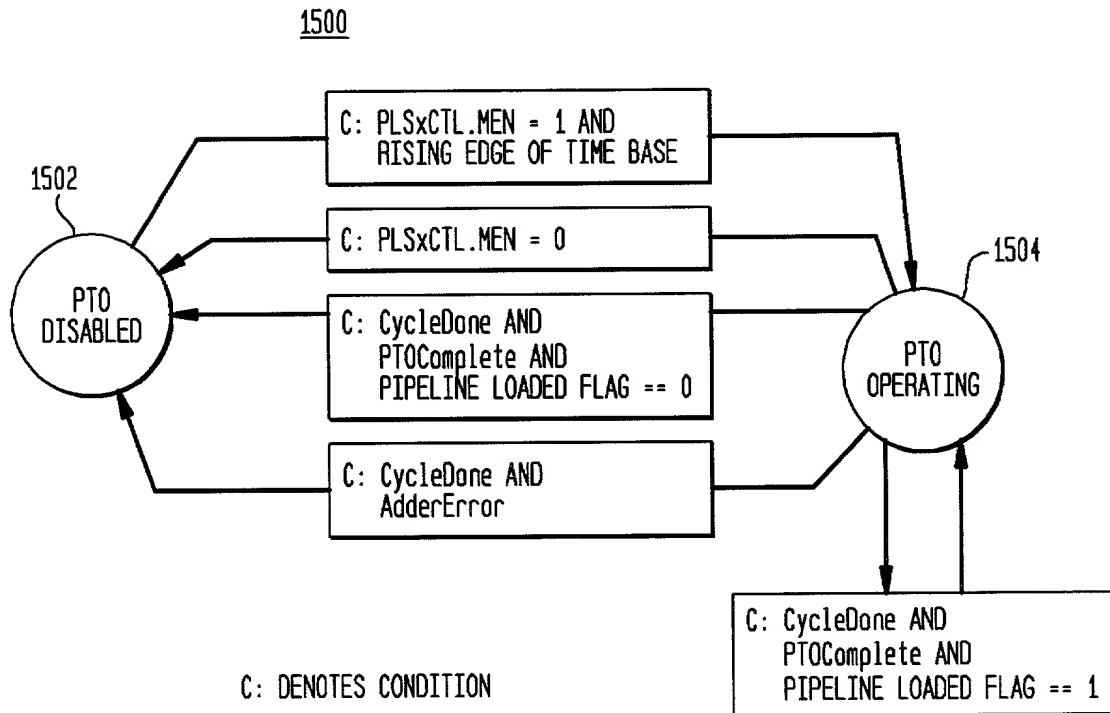


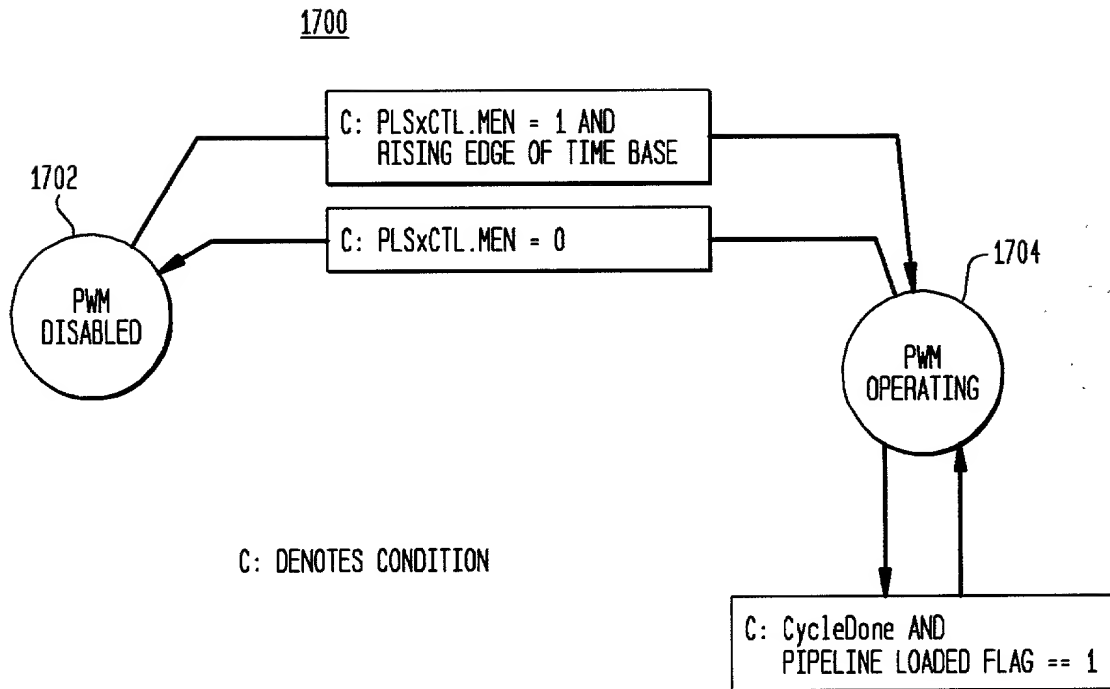
FIG. 16

```

cycle time counter = cycle time counter + 1;
IF (cycle time counter >= cycle time preset) THEN
BEGIN //this is the CycleDone event
    -----
    pulse counter = pulse counter + 1;
    IF (pulse counter >= pulse count preset) THEN
    BEGIN //this is the PTOComplete event
        -----
        assert INTxPLS signal, if PTOComplete interrupts are enabled;
        IF (pipeline loaded flag is set) THEN
        BEGIN
            -----
            transfer values from pipeline registers into operating registers;
            set pulse counter = 0;
            clear pipeline loaded flag;
            -----
        END
        ELSE //pipeline loaded flag is not set
        BEGIN
            -----
            GOTO PTO Disabled state; //disable the PLS block now
            -----
        END
        ENDIF
        -----
    END
    ELSE //not yet at PTOComplete
    BEGIN
        -----
        cycle time preset = cycle time preset + delta cycle time;
        IF (cycle time preset exceeds bounds) THEN
        BEGIN //this is the AdderError event
            -----
            assert INTxPLS signal, if AdderError interrupts are enabled;
            GOTO PTO Disabled state; //disable the PLS block now
            -----
        END
        ENDIF
        -----
    END
    ENDIF
    set cycle time counter = 0;
    -----
END
ENDIF

IF (cycle time counter >= (1/2 * cycle time preset)) THEN
    PLSxOUT = 0;
ELSE //output still in logic high portion of the current cycle
    PLSxOUT = 1;
ENDIF

```

FIG. 17**FIG. 18**

```

cycle time counter = cycle time counter + 1;
IF (cycle time counter >= cycle time preset) THEN
BEGIN //this is the CycleDone event
    -----
    IF (pipeline loaded flag is set) THEN
    BEGIN
        -----
        transfer values from pipeline registers into operating registers;
        clear pipeline loaded flag;
        -----
    END
    ENDIF
    set cycle time counter = 0;
    -----
END
ENDIF

IF (cycle time counter >= (pulse width preset)) THEN
    PLSxOUT = 0;
ELSE //output still in logic high portion of the current cycle
    PLSxOUT = 1;
ENDIF
  
```

FIG. 19

HIGH SPEED OPERATIONS				
INSTRUCTION	MNEMONIC & OPERAND(S)	DESCRIPTION	STL STATUS ELEMENT	VALID OPERANDS
PULSE TRAIN OUTPUT PROFILE	PTOP t, n	WHEN S0 = 1, THE PTO PROFILE SPECIFIED IN THE TABLE FOR OUTPUT n IS EXECUTED. ENO <- S0 * /e	STK, <CURRENT STEP>	ENABLE: S0 TABLE: VB, IB, QB, MB, (UI) SMB, SB, LB, *VD, *AC OUTPUT: KW (UI) 0 - 1

DEFINITION OF THE TABLE FOR PTO:

BYTE OFFSET	SEGMENT	DESCRIPTION OF TABLE ENTRIES
0		NUMBER OF PROFILE SEGMENTS (40 SEGMENTS MAXIMUM)
1		CURRENT STEP NUMBER BEING EXECUTED
2	#1	NUMBER OF STEPS (4 STEPS MINIMUM)
4		STARTING CYCLE TIME FOR THIS SEGMENT (2 TO 65535 μSEC)
6		CHANGE IN CYCLE TIME PER STEP (SIGNED VALUE) (0 TO 65535 μSEC)
8	#2	NUMBER OF STEPS (4 STEPS MINIMUM)
10		STARTING CYCLE TIME FOR THIS SEGMENT (2 TO 65535 μSEC)
12		CHANGE IN CYCLE TIME PER STEP (SIGNED VALUE) (0 TO 65535 μSEC)
⋮	⋮	⋮

FIG. 20

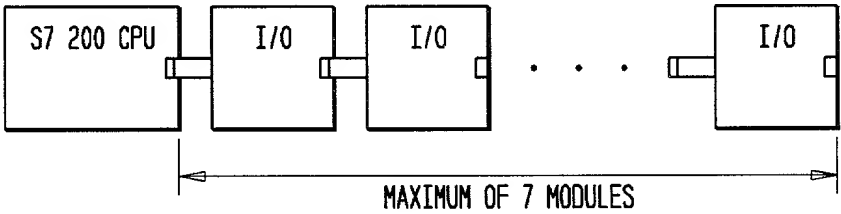


FIG. 21

SIGNAL NAME	USE
EMD	EXPANSION MODULE DATA - A BI-DIRECTIONAL SIGNAL USED TO COMMUNICATE THE ADDRESS AND DATA TO AND FROM THE MODULE.
EMC[1:0]	EXPANSION MODULE CLOCKS - ONE CLOCK IS USED TO ACCESS EXPANSION I/O EXTERNAL TO THE PLC, WHILE THE OTHER IS USED TO ACCESS I/O THAT IS LOCAL TO THE PLC THAT DOES NOT CONNECT DIRECTLY TO THE ASIC.
XA_OD	ADDRESS/OUTPUT DISABLE - A DUAL FUNCTION SIGNAL USED TO RESET THE STATE MACHINE IN THE MODULES ON THE FIRST CLOCK OF EACH ACCESS CYCLE (ACTIVE LOW) AND USED TO INDICATE OUTPUT DISABLE WHEN A FATAL ERROR HAS BEEN DETECTED (ACTIVE LOW FOR AN RC TIME CONSTANT).
EMDIR	EXPANSION MODULE DATA DIRECTION - THIS SIGNAL INDICATES THE DIRECTION OF DATA FLOW ON THE EMD SIGNAL LINE. 0 - DATA IS DRIVEN BY THE MODULE TO THE PLC 1 - DATA IS DRIVEN BY THE PLC TO THE MODULE
EMA[2:0]	EXPANSION MODULE ADDRESS - THESE SIGNALS ARE DAISY CHAINED FROM PLC TO MODULE TO MODULE. THE VALUE INPUT TO A MODULE BECOMES THAT MODULE'S ADDRESS. THE MODULE WILL OUTPUT ITS ADDRESS PLUS ONE TO THE NEXT MODULE. THE PLC DRIVES THESE SIGNALS TO 0 SO THAT THE MODULE CONNECTED TO THE PLC HAS THE ADDRESS OF 0.
+5V	5 VOLT POWER SUPPLY - TWO SIGNALS CARRY +5 VOLTS.
GND	POWER SUPPLY RETURN - TWO SIGNALS CARRY GROUND.

FIG. 23

REFERENCE DESIGNATOR	COMPONENT VALUE/TYPE	COMPONENT DESCRIPTION
IC1	74ABT125/74ABT126	TRI-STATE BUFFER
CR1	TVS 5.6V ZENER	DIODE
R1	4.7K ohm	RESISTOR
R2	110 ohm	RESISTOR
R3	220 ohm	RESISTOR
R4	22 ohm	RESISTOR
C1	0.1 μ F	CAPACITOR
C2	100pF	CAPACITOR

FIG. 24

SIGNAL NAME	DRIVING DEVICE	DRIVE LEVELS				RECEIVE LEVELS	
		V _{OL} (VOLTS) (MAX)	I _{OL} (mA) (MIN)	V _{OH} (VOLTS) (MAX)	I _{OH} (mA) (MIN)	V _{IL} (VOLTS) (MAX)	V _{IH} (VOLTS) (MIN)
EMA[2:0]	MODULE	0.5	3.0	2.4	-3.0	0.8	2.0
XA_OD	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
EMC	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
EMD	CPU	0.55	64.0	2.0	-32.0	0.8	2.0
	MODULE	0.55	64.0	2.0	-32.0	0.8	2.0

FIG. 25

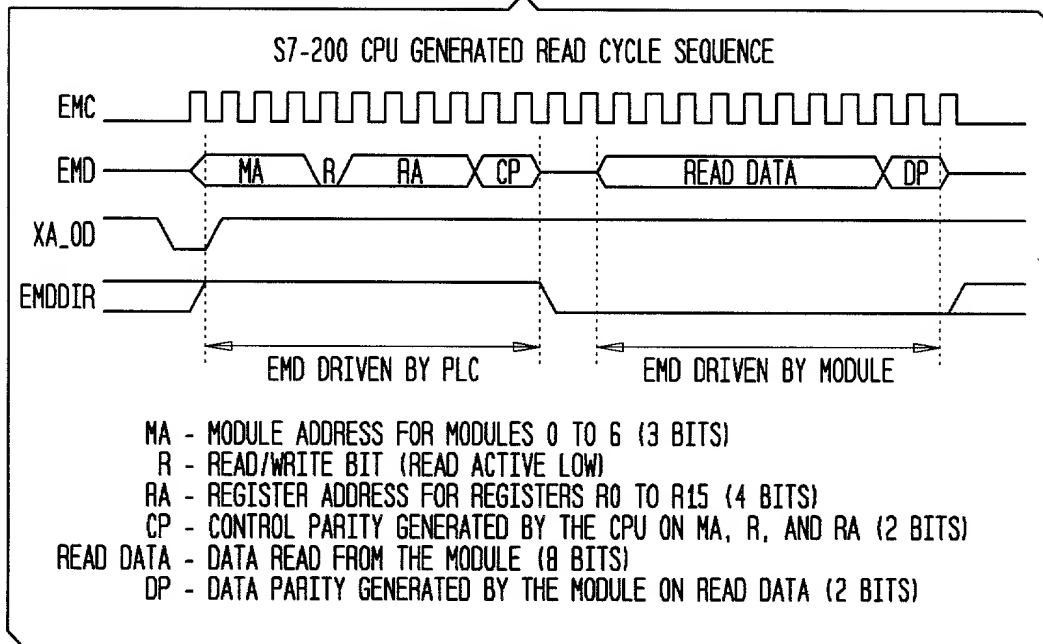


FIG. 26

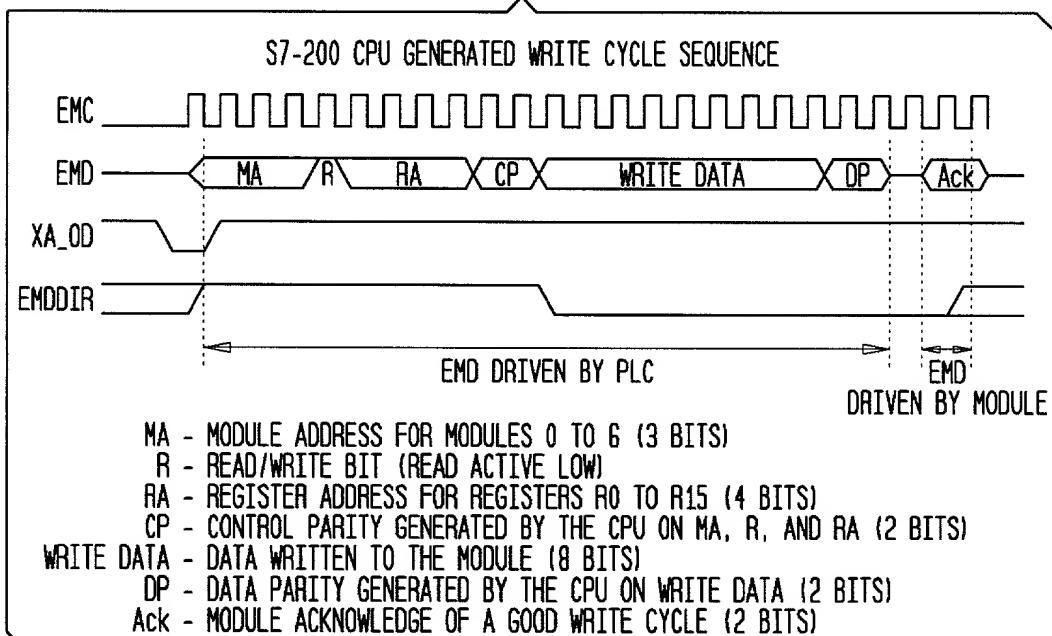


FIG. 27

	a	b	c	d	e
P0	BIT 0	BIT 2	BIT 4	BIT 6	BIT 7
P1	BIT 1	BIT 2	BIT 3	BIT 5	BIT 7
5 BIT ODD PARITY = !!(a xor b) xor (c xor d) xor e					

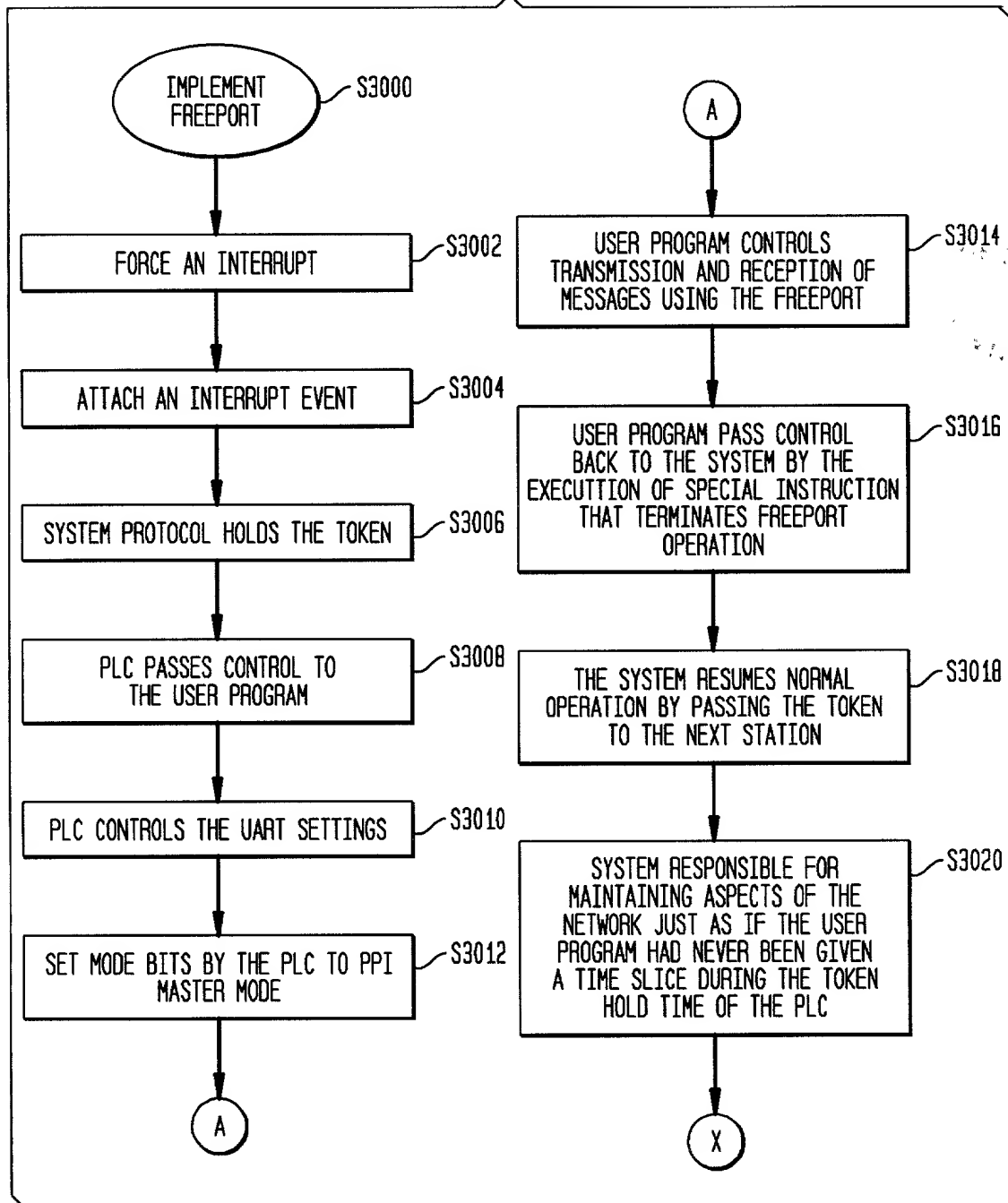
FIG. 28

PLC TYPE	PORT 0	PORT 1
CPU 212	256	-
CPU 214	256	-
CPU 215/216	256	256

FIG. 29

INTERRUPTS				
INSTRUCTION	MNEMONIC & OPERAND(S)	DESCRIPTION	STL STATUS ELEMENT	VALID OPERANDS
PASS TOKEN	PASS	USER PROGRAM HAS COMPLETED ITS USE OF THE TOKEN HOLD PERIOD AND IS RETURNING CONTROL TO THE SYSTEM.	STK	

FIG. 30A



[illegible][illegible]

FIG. 30C

SM BIT DEFINITION (READ/WRITE) (CONTINUED)									
SM BITS	DESCRIPTION								
SM87	<p>PORT 0: RECEIVE MESSAGE CONTROL BYTE, (CPU 212/214/216)</p> <div><div>MSB</div><div>7</div><div>LSB</div><div>0</div></div> <table><tr><td>en</td><td>sc</td><td>ec</td><td>il</td><td>c/m</td><td>tmr</td><td>bk</td><td>0</td></tr></table> <p>en: ENABLE/DISABLE RECEIVE MESSAGE BIT IS CHECKED EACH TIME THE RCV INSTRUCTION IS EXECUTED. IF THIS BIT IS A "0", THEN THE RECEIVE MESSAGE FUNCTION IS DISABLED. IF THIS BIT IS A "1", THEN THE RECEIVE MESSAGE FUNCTION IS ENABLED.</p> <p>sc: 0 - IGNORE SMB88; 1 - USE THE VALUE OF SMB88 TO DETECT START OF MESSAGE</p> <p>ec: 0 - IGNORE SMB89; 1 - USE THE VALUE OF SMB89 TO DETECT END OF MESSAGE</p> <p>il: 0 - IGNORE SMW90; 1 - USE THE VALUE OF SMW90 TO DETECT AN IDLE LINE CONDITION¹</p> <p>c/m: 0 - USE TIMER AS AN INTER-CHARACTER TIMER; 1 - USE TIMER AS A MESSAGE TIMER</p> <p>tmr: 0 - IGNORE SMW92; 1 - TERMINATE RECEIVE IF THE TIME PREIOD IN SMW92 IS EXCEEDED</p> <p>bk: 0 - IGNORE BREAK CONDITIONS; 1 - USE BREAK CONDITION AS START OF MESSAGE DETECTION</p> <p>¹BY SETTING THE sc AND bk BITS TO 0 AND THE en, il, c/m AND tmr BITS TO 1 WITH AN IDLE LINE TIMER VALUE OF ZERO, SMW92 WILL BE USED TO TIME OUT THE RCV INSTRUCTION WITHOUT RECEIVING ANY CHARACTERS. IF THE TIMER IS NOT USED (tmr = 0), THEN ANY CHARACTER RECEIVED WILL BE USED AS START OF MESSAGE.</p> <p>THE BITS OF THE MESSAGE INTERRUPT CONTROL BYTE ARE USED TO DEFINE THE CRITERIA BY WHICH THE MESSAGE IS IDENTIFIED. BOTH START OF MESSAGE AND END OF MESSAGE CRITERIA ARE DEFINED. TO DETERMINE THE START OF A MESSAGE EITHER OF TWO SETS OF LOGICALLY ANDDED START OF MESSAGE CRITERIA MUST BE TRUE AND MUST OCCUR IN SEQUENCE (IDLE LINE FOLLOWED BY START CHARACTER OR BREAK FOLLOWED BY START CHARACTER). TO DETERMINE THE END OF A MESSAGE THE ENABLED END OF MESSAGE CRITERIA ARE LOGICALLY Ored. THE EQUATIONS FOR START AND STOP CRITERIA ARE GIVEN BELOW:</p> <p style="margin-left: 40px;">Start of Message = $il * sc + bk * sc$</p> <p style="margin-left: 40px;">End of Message = $ec + tmr + \text{maximum character count reached}$</p> <p>NOTE: RECEIVE WILL AUTOMATICALLY BE TERMINATED BY AN OVERRUN OR A PARITY ERROR (IF ENABLED).</p>	en	sc	ec	il	c/m	tmr	bk	0
en	sc	ec	il	c/m	tmr	bk	0		
SM88	PORT 0: START OF MESSAGE CHARACTER. (CPU 212/214/216)								
SM89	PORT 0: END OF MESSAGE CHARACTER. (CPU 212/214/216)								
SM90 SM91	PORT 0: IDLE LINE TIME PERIOD GIVEN IN MILLISECONDS. THE FIRST CHARACTER RECEIVED AFTER THE IDLE LINE TIME HAS EXPIRED IS THE START OF A NEW MESSAGE. SM90 IS MSB. (CPU 212/214/216)								
SM92 SM93	PORT 0: INTER-CHARACTER/MESSAGE TIMER TIMEOUT VALUE GIVEN IN MILLISECONDS. IF THE TIME PERIOD IS EXCEEDED, THE RECEIVE MESSAGE IS TERMINATED. SM92 IS MSB. (CPU 212/214/216)								
SM94	PORT 0: MAXIMUM NUMBER OF CHARACTERS TO BE RECEIVED (1 TO 255 BYTES) (CPU 212/214/216)								

FIG. 30D

SM BIT DEFINITION (READ/WRITE) (CONTINUED)									
SM BITS	DESCRIPTION								
SM187	<p>PORT 1: MESSAGE INTERRUPT CONTROL BYTE, (CPU 216 ONLY)</p> <div style="display: flex; justify-content: space-between; align-items: center;"><div>MSB 7</div><div>LSB 0</div></div> <table border="1" style="margin: 10px auto; text-align: center; border-collapse: collapse;"><tr><td style="padding: 2px 5px;">en</td><td style="padding: 2px 5px;">sc</td><td style="padding: 2px 5px;">ec</td><td style="padding: 2px 5px;">il</td><td style="padding: 2px 5px;">c/m</td><td style="padding: 2px 5px;">tmr</td><td style="padding: 2px 5px;">bk</td><td style="padding: 2px 5px;">0</td></tr></table> <p>en: ENABLE/DISABLE RECEIVE MESSAGE BIT IS CHECKED EACH TIME THE RCV INSTRUCTION IS EXECUTED. IF THIS BIT IS A "0", THEN THE RECEIVE MESSAGE FUNCTION IS DISABLED. IF THIS BIT IS A "1", THEN THE RECEIVE MESSAGE FUNCTION IS ENABLED.</p> <p>sc: 0 - IGNORE SMB188; 1 - USE THE VALUE OF SMB188 TO DETECT START OF MESSAGE</p> <p>ec: 0 - IGNORE SMB189; 1 - USE THE VALUE OF SMB189 TO DETECT END OF MESSAGE</p> <p>il: 0 - IGNORE SMW190; 1 - USE THE VALUE OF SMW190 TO DETECT AN IDLE CONDITION¹</p> <p>c/m: 0 - USE TIMER AS AN INTER-CHARACTER TIMER; 1 - USE TIMER AS A MESSAGE TIMER</p> <p>tmr: 0 - IGNORE SMW192; 1 - TERMINATE RECEIVE IF THE TIME PREIOD IN SMW192 IS EXCEEDED</p> <p>bk: 0 - IGNORE BREAK CONDITIONS; 1 - USE BREAK CONDITION AS START OF MESSAGE DETECTION</p> <p>¹BY SETTING THE sc AND bk BITS TO 0 AND THE en, il, c/m AND tmr BITS TO 1 WITH AN IDLE LINE TIMER VALUE OF ZERO, SMW192 WILL BE USED TO TIME OUT THE RCV INSTRUCTION WITHOUT RECEIVING ANY CHARACTERS. IF THE TIMER IS NOT USED (tmr = 0), THEN ANY CHARACTER RECEIVED WILL BE USED AS START OF MESSAGE.</p> <p>THE BITS OF THE MESSAGE INTERRUPT CONTROL BYTE ARE USED TO DEFINE THE CRITERIA BY WHICH THE MESSAGE IS IDENTIFIED. BOTH START OF MESSAGE AND END OF MESSAGE CRITERIA ARE DEFINED. TO DETERMINE THE START OF A MESSAGE EITHER OF TWO SETS OF LOGICALLY ANDDED START OF MESSAGE CRITERIA MUST BE TRUE AND MUST OCCUR IN SEQUENCE (IDLE LINE FOLLOWED BY START CHARACTER OR BREAK FOLLOWED BY START CHARACTER). TO DETERMINE THE END OF A MESSAGE THE ENABLED END OF MESSAGE CRITERIA ARE LOGICALLY Ored. THE EQUATIONS FOR START AND STOP CRITERIA ARE GIVEN BELOW:</p> <div style="margin-left: 40px;"><p>Start of Message = $il * sc + bk * sc$</p><p>End of Message = $ec + tmr + \text{maximum character count reached}$</p></div> <p>NOTE: RECEIVE WILL AUTOMATICALLY BE TERMINATED BY AN OVERRUN OR A PARITY ERROR (IF ENABLED).</p>	en	sc	ec	il	c/m	tmr	bk	0
en	sc	ec	il	c/m	tmr	bk	0		
SM188	PORT 1: START OF MESSAGE CHARACTER. (CPU 216 ONLY)								
SM189	PORT 1: END OF MESSAGE CHARACTER. (CPU 216 ONLY)								
SM190 SM191	PORT 1: IDLE LINE TIME PERIOD GIVEN IN MILLISECONDS. THE FIRST CHARACTER RECEIVED AFTER THE IDLE LINE TIME HAS EXPIRED IS THE START OF A NEW MESSAGE. SM190 IS MSB. (CPU 216 ONLY)								
SM192 SM193	PORT 1: INTER-CHARACTER/MESSAGE TIMER TIMEOUT VALUE GIVEN IN MILLISECONDS. IF THE TIME PERIOD IS EXCEEDED, THE RECEIVE MESSAGE IS TERMINATED. SM192 IS MSB. (CPU 216 ONLY)								
SM194	PORT 1: MAXIMUM NUMBER OF CHARACTERS TO BE RECEIVED (1 TO 255 BYTES) (CPU 216 ONLY)								

FIG. 30E

PLC TYPE	PORT 0			PORT 1		
	NUMBER OF CONNECTIONS		BUFFER SIZE (BYTES)	NUMBER OF CONNECTIONS		BUFFER SIZE (BYTES)
	TOTAL	RESERVED		TOTAL	RESERVED	
CPU 221	4	1 - PG 1 - OP	128/256	-	-	-
CPU 222	4	1 - PG 1 - OP	128/256	-	-	-
CPU 224	4	1 - PG 1 - OP	128/256	-	-	-
CPU 226	4	1 - PG 1 - OP	128/256	4	1 - PG 1 - OP	128/256

FIG. 31

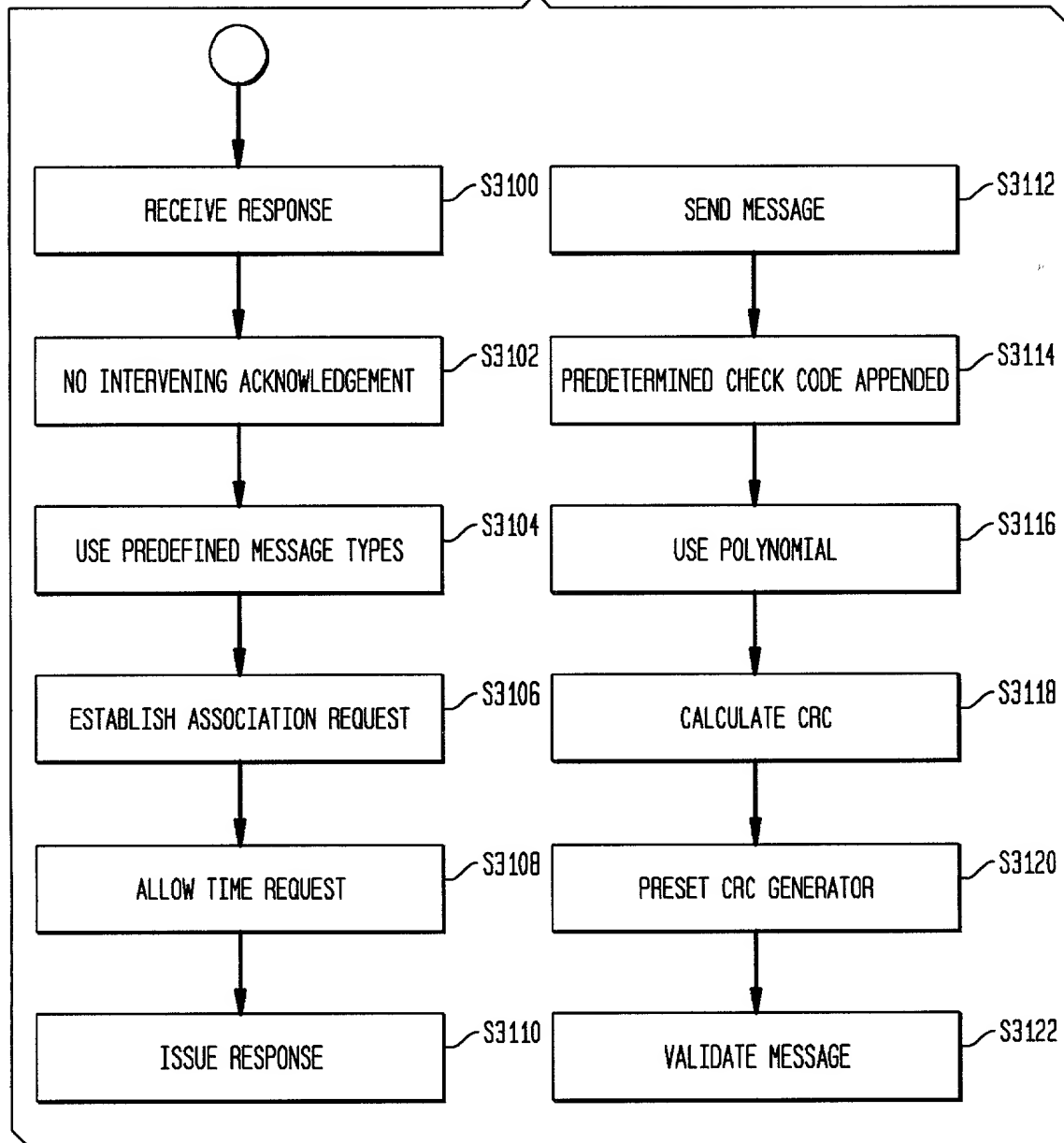


FIG. 32A

PROGRAM CONTROL FUNCTIONS				
INSTRUCTION	MNEMONIC & OPERAND(S)	DESCRIPTION	STL STATUS ELEMENT	VALID OPERANDS
HIDE	HIDE n, a, p	THE HIDE LABEL MARKS THE START OF A BLOCK OF n INSTRUCTIONS THAT ARE ENCRYPTED USING THE PASSWORD, p, WHEN a IS NON-ZERO.	STK, SMB1	ENABLE: NONE n: KW(2 BYTES) (UI) a: KW(2 BYTES) (UI) p: KD(4 BYTES) (UI)

FIG. 32B

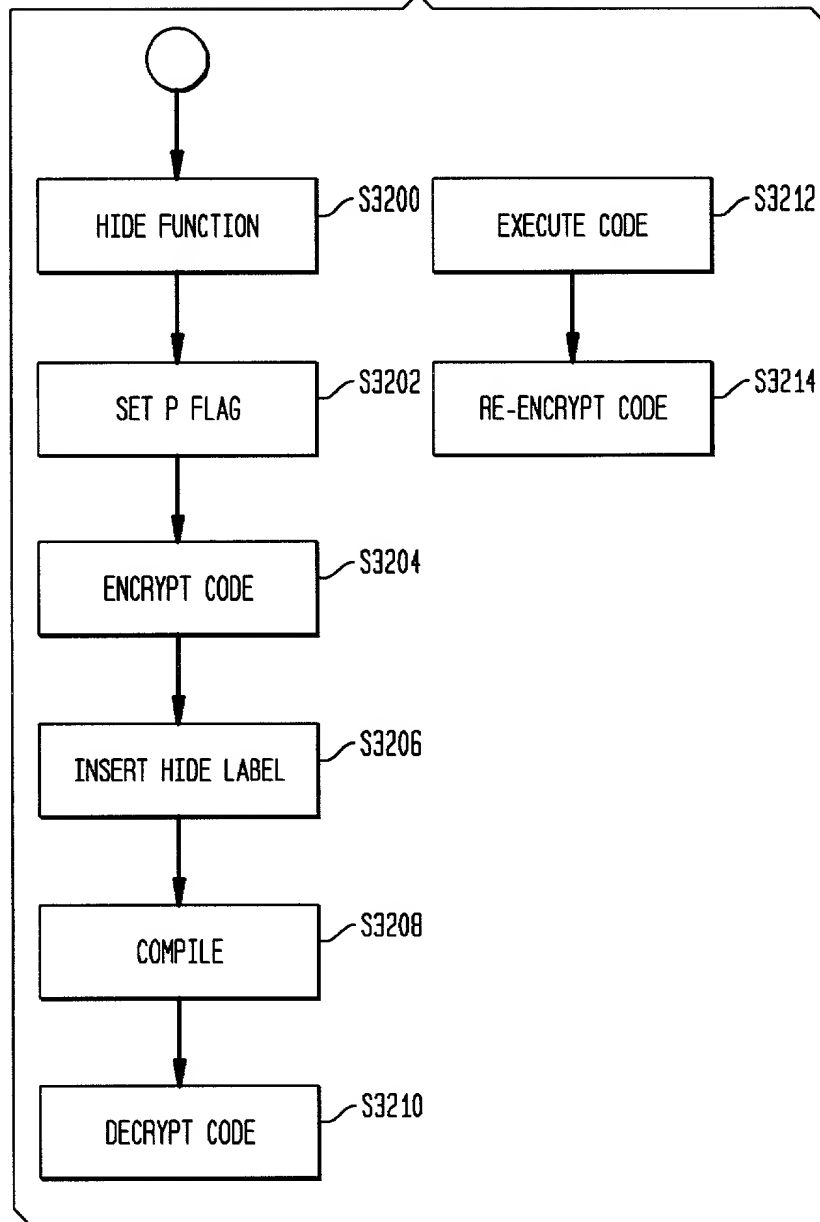


FIG. 33A

PROGRAM CONTROL FUNCTIONS				
INSTRUCTION	MNEMONIC & OPERAND(S)	DESCRIPTION	STL STATUS ELEMENT	VALID OPERANDS
SYSTEM FUNCTION CALL	SFC f,s, #_parms, p0,p1,p2,...	WHEN S0 = 1, EXECUTE THE SYSTEM FUNTION IDENTIFIED BY f AND THE SUB- FUNCTION IDENTIFIED BY s.	STK, SMB1, <p0>, <p1>, <p2>,...	<p>ENABLE: S0 f: KW (UI) 0-65536 s: KW (UI) 0-65536</p> <p>#parms: KB (UI) 0-16</p> <p>p0,...: Bit,Byte,Word, Dword Bit V,I,Q,M,SM,S, T,C,L Byte VB,IB,QB,MB, SMB,SB,KB,LB Word VW,T,C,IW, QW,MW,SMW, SW,LW,KW Dword VD,ID,QD,MD, SMD,SD,HC, LD,KD,&VB, &IB,&QB,&MB, &T,&C,&SB</p> <p>NOTE: CONSTANTS AND ADDRESS POINTER SPECIFICATIONS ARE NOT ALLOWED FOR OUTPUT OR INPUT/OUTPUT PARAMETERS.</p>

FIG. 33B

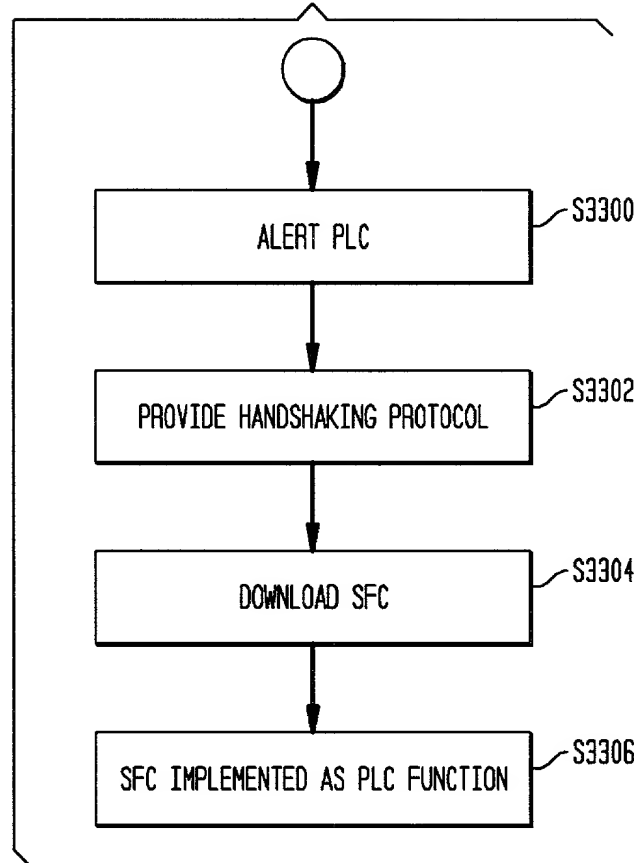


FIG. 34A

INSTRUCTION ADDRESS OF WINDOW START								2 BYTES
LENGTH OF STATUS DATA FOR 1ST INSTRUCTION								1 BYTE
V	ENO	SCR	S4	S3	S2	S1	S0	1 BYTE
STL STATUS DATA								n BYTES
LENGTH OF STATUS DATA FOR 2ND INSTRUCTION								1 BYTE
V	ENO	SCR	S4	S3	S2	S1	S0	1 BYTE
STL STATUS DATA								n BYTES
⋮								

FIG. 34B

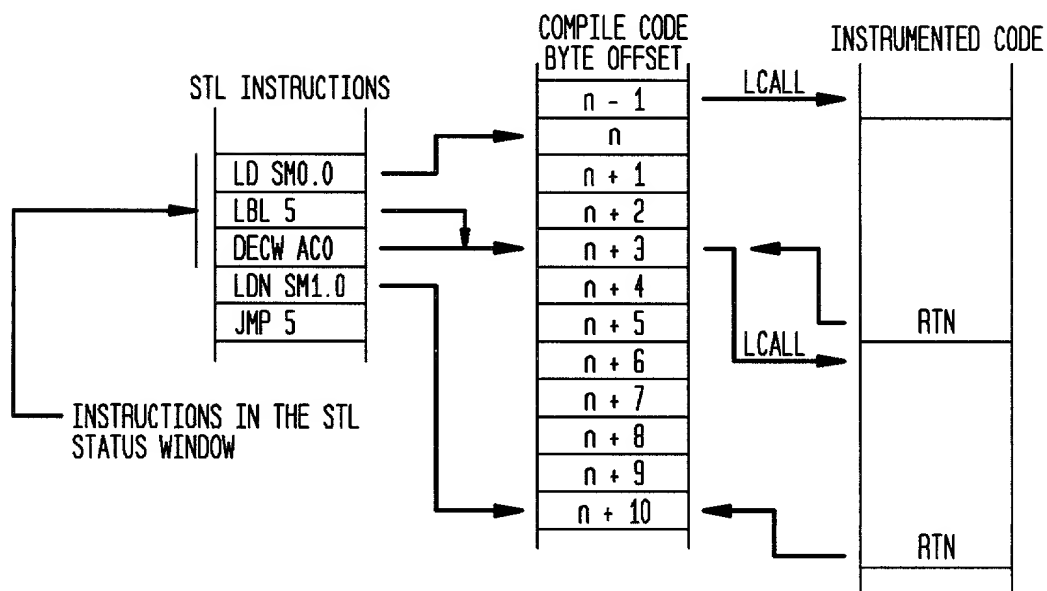


FIG. 34C

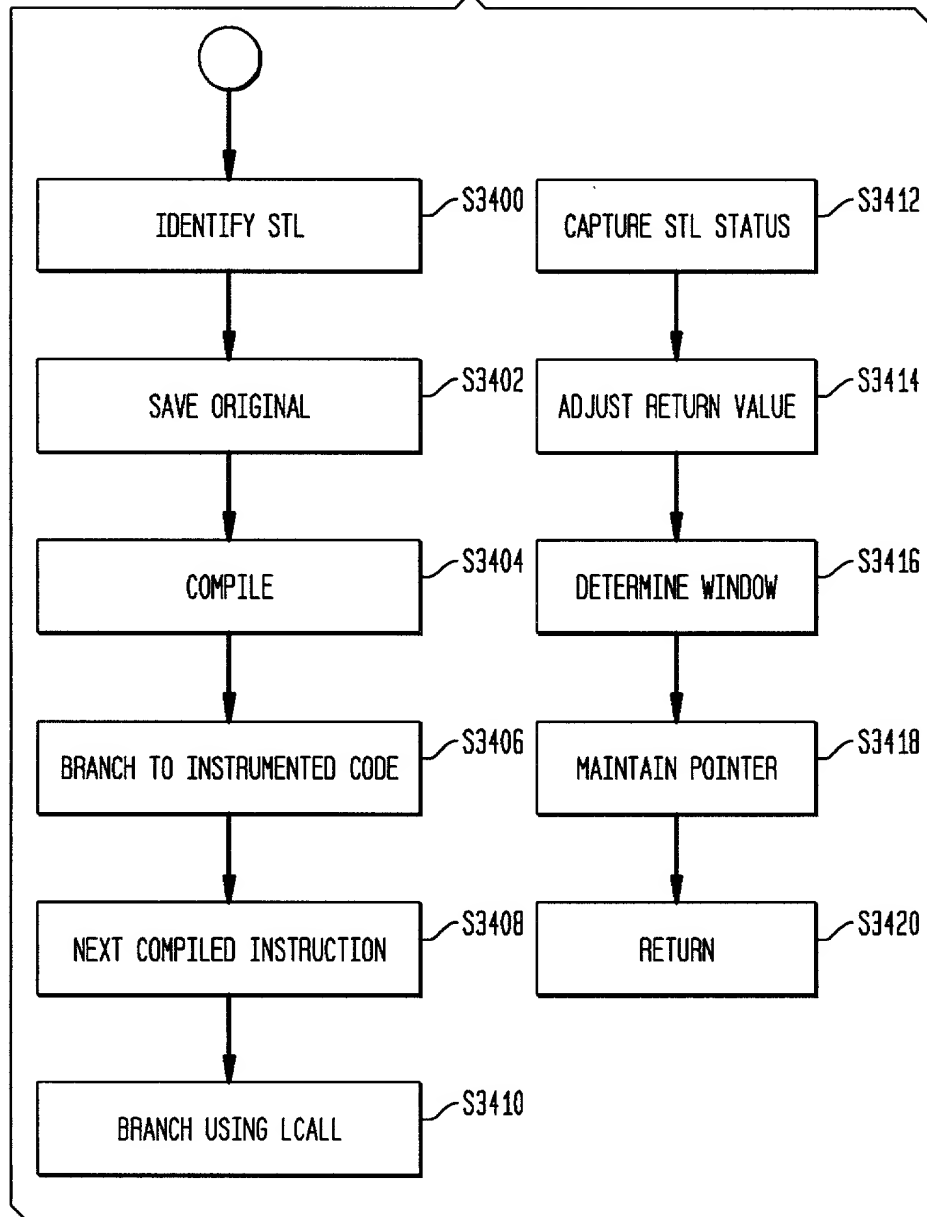


FIG. 34C

	BASIC INSTR.	COMPILED TO	BYTES	CYCLES
1	LD	RLC A MOV C,<BIT>	1 2	1 1
2	LDN	RLC A MOV C,<BIT> CPL C	1 2 1	1 1 1
3	A	ANL C,<BIT> NOP	2 1	2 1
4	AN	ANL C,/ <BIT> NOP	2 1	2 1
5	O	ORL C,<BIT> NOP	2 1	2 1
6	ON	ORL C,/ <BIT> NOP	2 1	2 1
7	=	MOV <BIT>,C NOP	2 1	2 1
8	ALD	ANL C,ACC.0 RR A	2 1	2 1
9	OLD	ORL C,ACC.0 RR A	2 1	2 1
10	NOT	CPL C NOP NOP	1 1 1	1 1 1
11	LPS	RL A MOV ACC.0,A	1 2	1 2
12	LPP	RRC A NOP NOP	1 1 1	1 1 1
13	LAD	MOV C,ACC.0	2	2
14	RET	RET NOP NOP	1 1 1	2 1 1
15	INT	CLR A NOP NOP	1 1 1	1 1 1
16	END	JNC \$+0 RET	2 1	2 2
17	CRETI	JNC \$+0 RET	2 1	2 2

FIG. 35A

HOUSING	HEIGHT	WIDTH	DEPTH
CPU 221	80 mm	90 mm	62 mm
CPU 222	80 mm	90 mm	62 mm
CPU 224	80 mm	120.5 mm	62 mm
CPU 226	80 mm	190 mm	62 mm
8 POINT I/O MODULE	80 mm	46 mm	62 mm
16 POINT I/O MODULE	80 mm	71.2 mm	62 mm
32 POINT I/O MODULE	80 mm	125 mm	62 mm
INTELLIGENT MODULE	80 mm	90 mm	62 mm

FIG. 35B

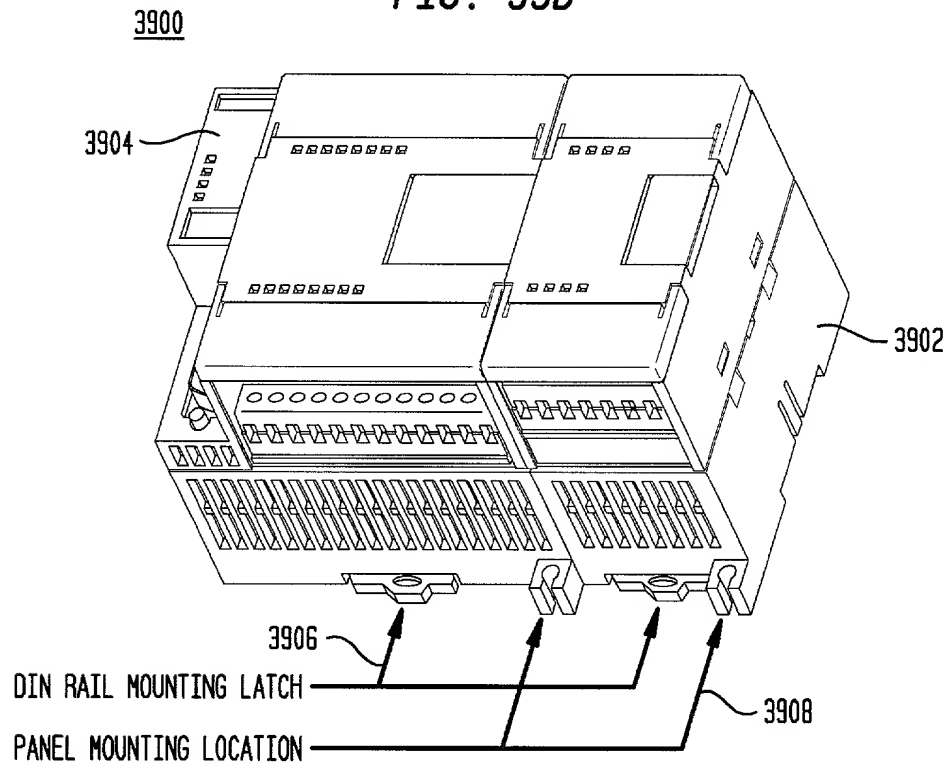


FIG. 36

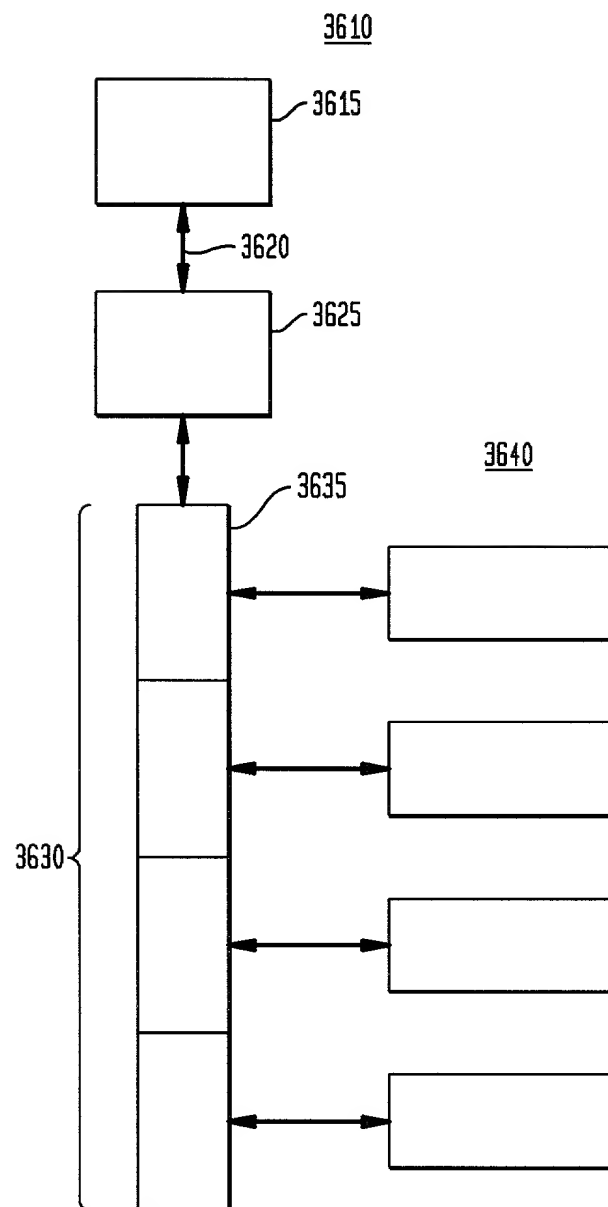


FIG. 37A

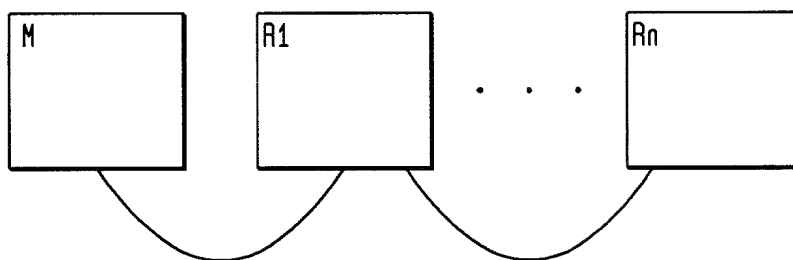


FIG. 37B

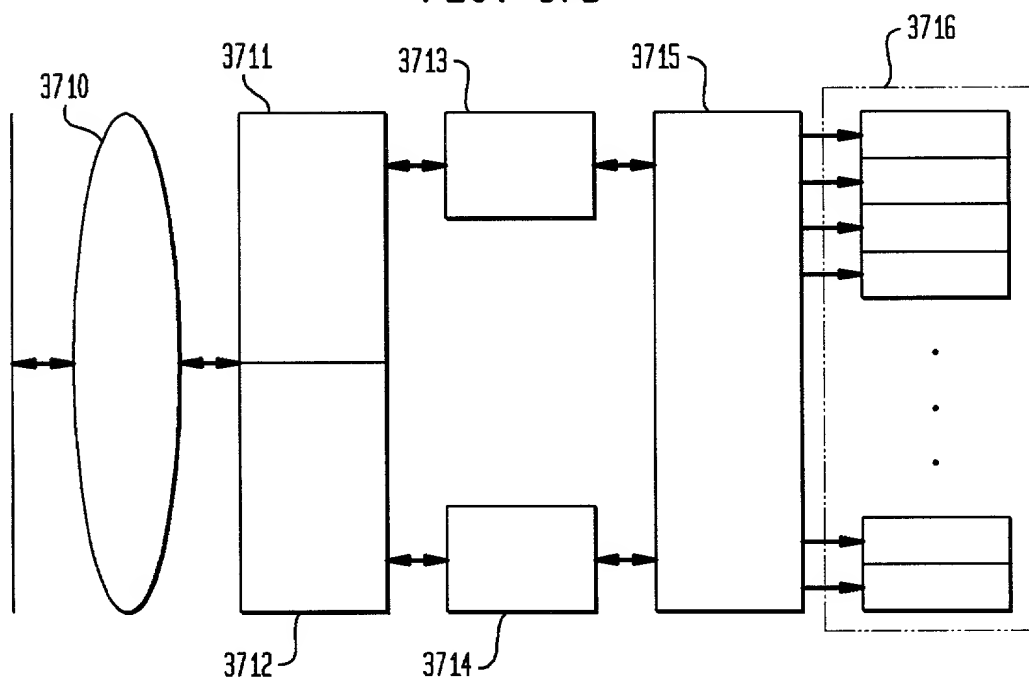


FIG. 37A

FIG. 38

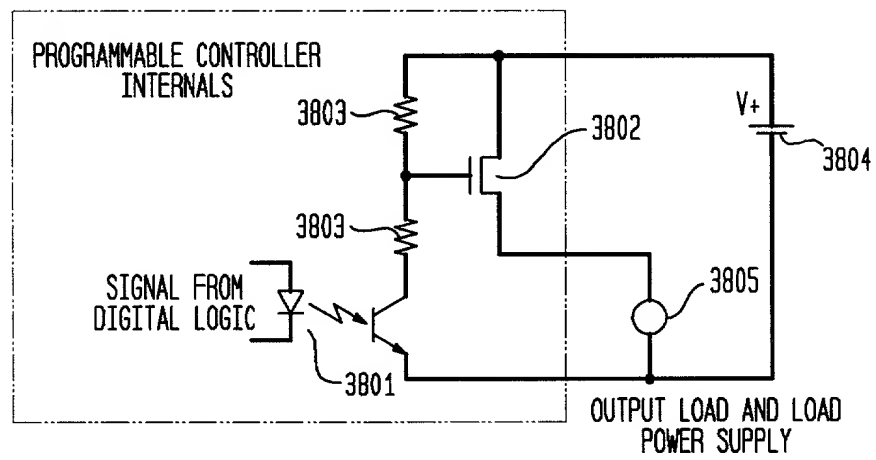


FIG. 39

